第2章

パソコン用ディスプレイに表示してみよう

画面サイズに合わせてアナログRGB出力回路を作る

江崎雅康/岩田正雄

今回使用するパソコン用ディスプレイのサイズはVGA(640×480ピクセル)である．この仕様に合わせて，FPGAで信号を作る方法を紹介する．　（編集部）

1．アナログRGBインターフェース信号

アナログRGB信号は，現在もパソコン表示装置の標準インターフェース信号として使われています．コネクタは写真1の15ピン・シュリンクDサブ・コネクタが使われています．

表示デバイスはCRTから液晶に代わりましたが，信号仕様は基本的に変わっていません．アナログRGB出力信号は表1に示すように，3原色アナログ画像信号である，

- R(赤)
- G(緑)
- B(青)

および，

- 水平同期信号(Hsync)
- 垂直同期信号(Vsync)

の5本で構成されています．

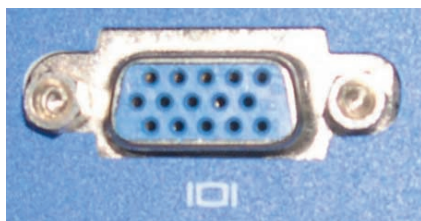



写真1　15ピン・シュリンクDサブ・コネクタ

この5種類の信号で市販のパソコン用表示装置にカラー画像を表示します．

画像ベースボードCQ-SP3EDWは付属基板上のSpartan-3Eを使ってこのアナログRGB信号を生成します．図1はその回路図です．

付属基板搭載のFPGAから出力されるディジタルの画像データ信号である，

- R(赤)　LR[5, 4, 3, 2, 1, 0, −1, −2]
- G(緑)　LG[5, 4, 3, 2, 1, 0, −1, −2]
- B(青)　LB[5, 4, 3, 2, 1, 0, −1, −2]

は，A-Dコンバータ「ADV7125」でアナログ画像信号に変

表1　アナログRGBインターフェースのシュリンク15ピン・コネクタの信号配列

画像データ信号はアナログ値．水平/垂直同期信号はディジタル信号．パソコンのアナログRGB画面表示の基本はこの5本の信号．

ピン番号	機　能
1	画像データ(R：赤)
2	画像データ(G：緑)
3	画像データ(B：青)
4	−
5	−
6	グラウンド(GND)
7	グラウンド(GND)
8	グラウンド(GND)
9	−
10	グラウンド(GND)
11	グラウンド(GND)
12	−
13	水平同期信号(Hsync)
14	垂直同期信号(Vsync)
15	−

KeyWord　アナログRGBインターフェース，15ピンDサブ・コネクタ，画像用D-Aコンバータ，ADV7125，垂直同期信号繰り返し周波数，水平走査線数，ピクセル・クロック

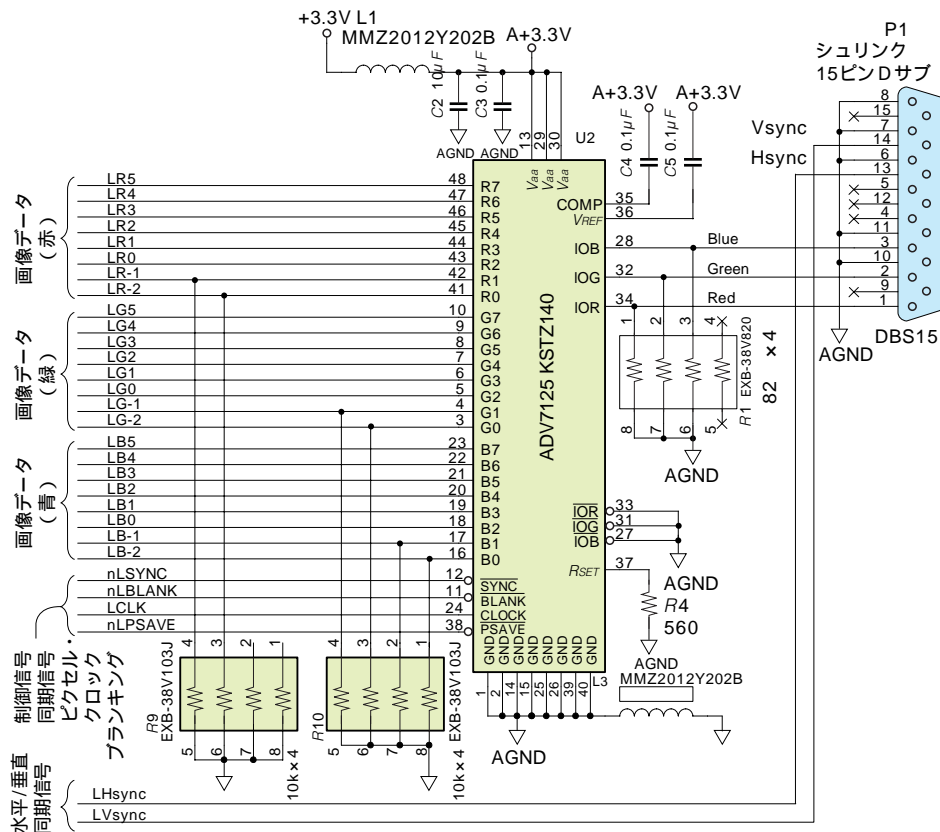



図1
アナログRGB出力回路
水平/垂直同期信号はFPGAから出力される信号をシュリンク15ピン・コネクタに直結．画像データ信号はADV7125でアナログ信号に変換．

換されます．

図では画像信号入力は8ビットで表示されていますが，付属基板の場合はピン数の制約で，

LR[5, 4, 3, 2, 1, 0, −1, −2]

LG[5, 4, 3, 2, 1, 0, −1, −2]

LB[5, 4, 3, 2, 1, 0, −1, −2]

の各6ビットのみ有効です．

残りの各2ビットはプルダウンしています．将来，付属基板を208ピンのSpartan-3E搭載基板に交換した場合は，フル8ビットのアナログ出力もできるようにしました．

ADV7125に入力している

- ピクセル・クロック　LCLK
- ブランキング信号（負論理）　nLBLANK
- 複合同期信号（負論理）　nLSYNC

は，画像データの変換に必要な信号です．

垂直/水平同期信号は直接，画像ベースボード上の15ピン・シュリンクDサブ・コネクタに出力しています．

P1の15ピンのシュリンクDサブ・コネクタに市販のパソコン用表示装置（CRT，液晶）の信号ケーブルを接続すると，VGAカラー表示ができます．

2. VGA画像信号のタイミング設計

図2は今回設計したアナログRGB信号のVGA（640×480ピクセル）画面タイミング図，**表2**はタイミング・パラメータです．

このアナログRGB信号のタイミング仕様は家庭用テレビから発達してきた経緯を反映しています．テレビの同期信号は，次の通りです．

- 垂直同期信号繰り返し周波数　60Hz
- 1画面の水平走査線数　525本インターレース

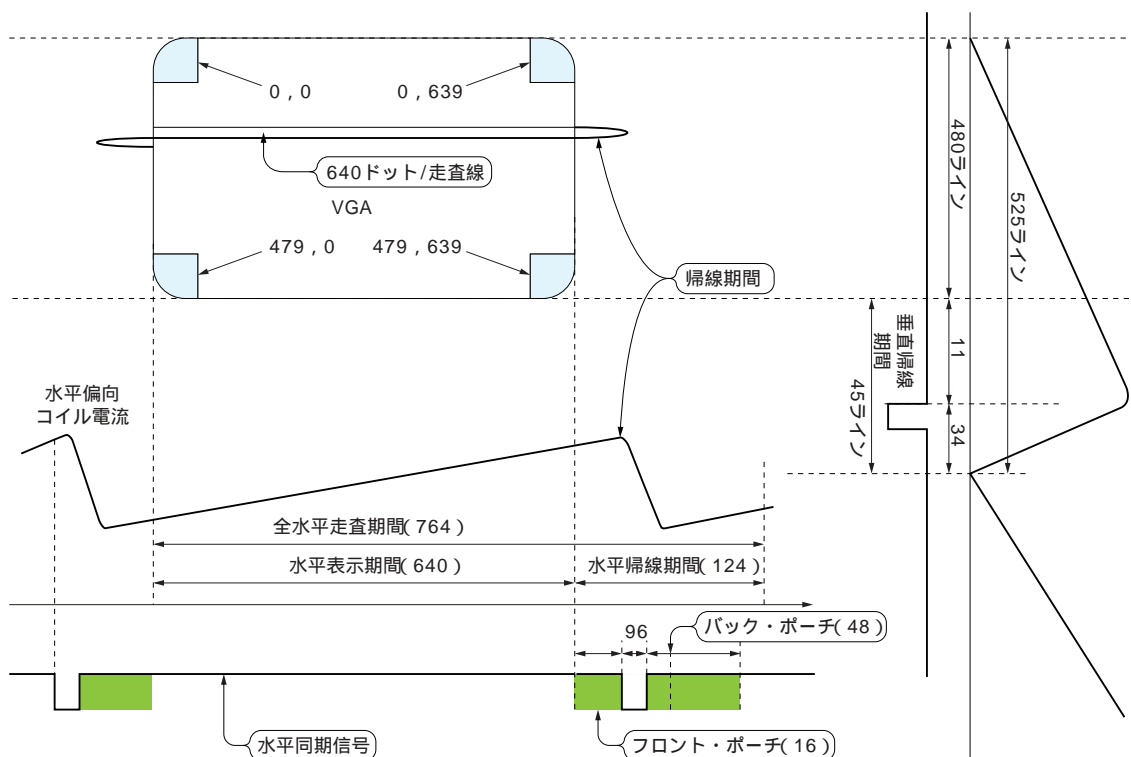
今回の設計では表示画面の同期信号は液晶モジュールおよびディジタルCMOSカメラとの同期設計を考えて，

- ピクセル・クロック　24MHz
- 垂直繰り返し周波数　60Hz
- 1画面の水平走査線数　525本ノン・インターレース

で設計することにしました．

これらのパラメータを液晶モジュールやディジタルCMOSカメラと一致させておくと，

- 液晶表示モジュールと同期信号および出力データ線を共

図2
VGA画面の同期信号タイミング設計
ピクセル・クロック24MHzをベースに水平/垂直同期信号をつくる．表示期間はVGA(640×480ピクセル)サイズ．

2

通にできる．

- カメラの画像入力信号をスルー表示させる時のタイミング設計が容易．
- カメラ画像をメモリに記憶させる場合でも垂直繰り返し周波数やピクセル・クロック周波数を一致させるとタイミング設計が容易．

などのメリットがあります．

図2の同期信号タイミング図に示すように，CRTディスプレイは陰極線によって作られる光のポイントを左右に走査(水平走査)することにより水平1ライン分の画像を表示します．光の点が左端から右端まで走査し，再び左端まで帰ってくる期間を「全水平走査期間」と呼びます．そのうち画面表示期間は「水平表示期間」，左端に帰る期間は「水平帰線期間」と呼びます．

VGA画面の場合，水平表示期間は640ピクセルです．残りの期間は走査線が右端から左端に戻る時間で，水平同期信号を境にしてそれぞれ「フロント・ポーチ」，「バック・ポーチ」と呼ばれています．

VGA画面の同期信号の標準値は**表2**のように決めましたが，繰り返し周波数は標準値から多少ずれていてもかまいません．表示装置の同期周波数に引き込まれて安定した画

表2　同期信号のタイミング・パラメータ
今回の設計に用いたパラメータはディジタルCMOSカメラ，液晶モジュールとの同期設計を容易にすることを考慮して決定した．

パラメータ	水平走査		垂直走査	
	時間	クロック数	時間	走査線数
周期	31.8μs	764	16.7ms	525
表示期間	26.7μs	640	15.3ms	480
フロント・ポーチ	667 ns	16	318μs	10
同期パルス幅	2.67μs	64	63.7μs	2
バック・ポーチ	1.83μs	44	1.05ms	33

像が表示されるからです．

ピクセル・クロックは液晶表示モジュールおよびCMOSカメラの標準値に合わせて24MHz，FPGAの基本クロックはその2倍の48MHzとしました．

繰り返し周波数を60Hz，1画面あたりの総水平走査線数を525本とすると，水平走査線あたりのピクセル・クロック数は，次のようになります．

24,000,000÷60÷525　761.90

今回の設計ではCMOSカメラの仕様に合わせて，

- ピクセル・クロック数/Hsync：764
- 水平走査線数/1フレーム：525

としました．

その結果，垂直繰り返し周波数は，次のようになります．

24,000,000 ÷ 764 ÷ 525 = 59.835…

ピクセル・クロック24MHzは，付属基板上の水晶発振器を取り替えないと変更できません．しかし，その他のタイミング・パラメータはVHDL記述によって自由に変えられます．

3．3チャネル画像用D-AコンバータADV7125

図3(a)は画像用D-Aコンバータ「ADV7125」(米国Analog Devices社)の内部ブロック図，図3(b)はパッケージ信号配列です．

ADV7125には表3に示す4種類のクロック速度オプションがあります．今回の画像ベースボードは24MHzですから，いちばん低コストの「ADV7125KST50」を使いました．

ADV7125には8ビット高速D-Aコンバータが3チャネル入っています．画像ベースボードの回路では，8ビットのうち上位6ビットをFPGAのデータ信号に接続し，下位2ビットは抵抗でプルダウンし00に設定しています．

8ビットの画像データは図4に示すようにピクセル・クロックCLOCKの立ち上りエッジでサンプリングされアナログ信号に変換されます．

アナログ出力信号の伝送インピーダンス75Ωとマッチングさせるため，ADV7125の出力端IOR，IOG，IOBを75Ωでプルダウンします．画像ベースボードは集合抵抗を使

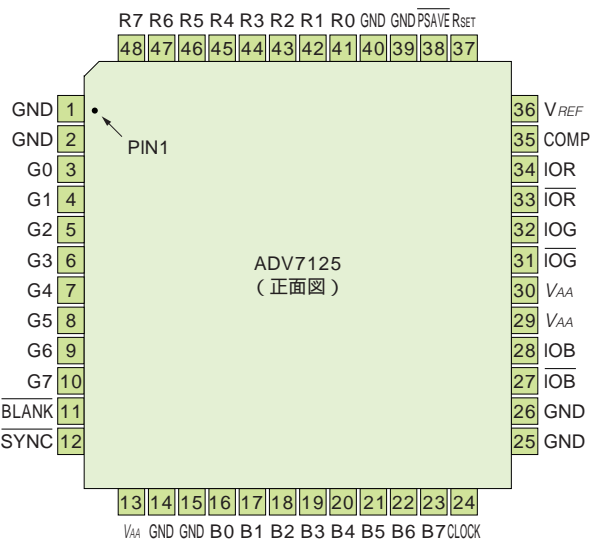
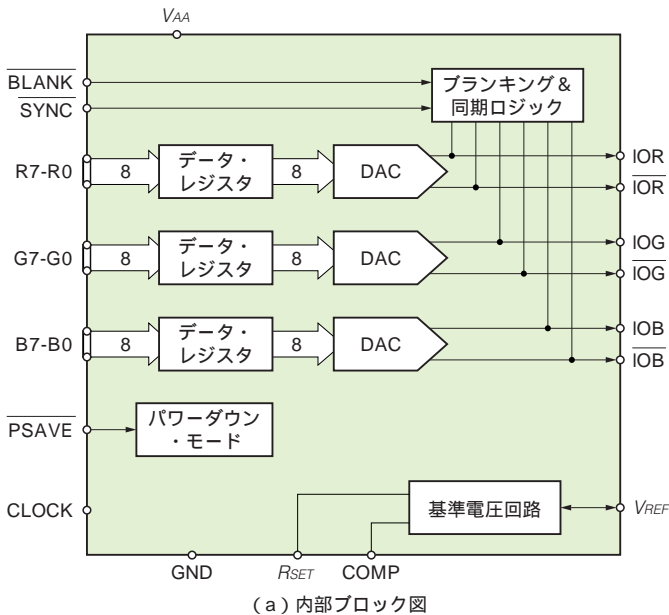



図3　3チャネル8ビット高速ビデオD-AコンバータADV7125

赤(R)，緑(G)，青(B)の3原色用に8ビット高速D-Aコンバータが3チャネル内蔵されている．入力画像データはピクセル・クロック(CLOCK)に同期してデータ・レジスタにセットされてアナログ信号に変換される．

表3　3チャネル8ビット高速ビデオD-AコンバータADV7125のクロック速度オプション

Analog Devices社のD-AコンバータADV7125には50MHz～330MHzまで4ランクのスピード・オプションがある．

品　番	クロック速度オプション
ADV7125KST50	50MHz
ADV7125KST140	140MHz
ADV7125JST240	240MHz
ADV7125JST330	330MHz

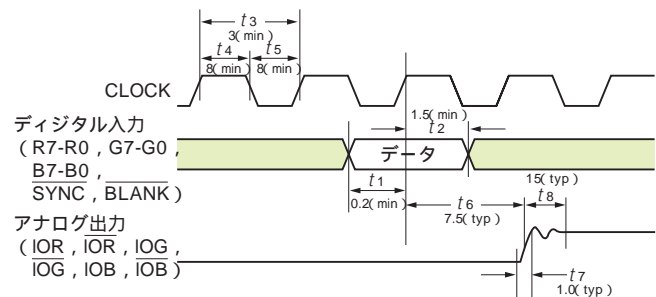



図4　3チャネル8ビット高速ビデオD-AコンバータADV7125のタイミング(単位：ns)

入力画像データはピクセル・クロック(CLOCK)の立ち上がりエッジに同期してデータ・レジスタにセットされる．

用し，その調達の都合により82Ωを使っています．

ADV7125のR_{SET}端子とグラウンド間には画像信号のフルスケールを調節するための抵抗を接続します．

同期信号$\overline{\text{SYNC}}$，ブランキング信号$\overline{\text{BLANK}}$は，図5に示すように，アナログRGB出力の信号レベルをコントロールします．

ADV7125の同期信号端子$\overline{\text{SYNC}}$に複合同期信号を入力すると，図5に示すようにG（緑）信号に同期信号が重畳されます．この必要がないときは同期信号端子$\overline{\text{SYNC}}$は“L”に固定します．

4．同期信号などの生成

図6はVGA制御回路のブロック図です．Spartan-3Eボードに実装した発振器の48MHzクロックを分周した24MHzをピクセル・クロックとします．

これを基準に表4のタイミング設計で水平/垂直同期信号，ブランキング信号を作ります．24MHzを764分周して水平同期信号，さらに525分周して垂直同期信号を作ります．

この同期カウンタのタイミングをベースに，

- 水平同期信号　　Hsync
- 垂直同期信号　　Vsync
- ブランキング信号　　nLBLANK

など必要な同期信号を作ります．ピクセル・クロック（LCLK）はそのまま出力します．

この同期カウンタの表示期間（640×480ピクセル）にあわせて画像データを出力します．フレーム・メモリから画像データを読み出す場合はカウンタ値からアドレスを算出します．

リスト1は表3に従って記述したVHDLのソースです．①では内部信号，

24MHzクロック信号　　clk24
水平カウンタ　　h_counter
垂直カウンタ　　v_counter

を定義しています．

水平カウンタは0～763，垂直カウンタは0～524まで変化するので，どちらも10ビット必要です．

②は24MHzクロックの生成，③は水平/垂直カウンタを記述しています．④は水平同期信号Hsync，垂直同期信号Vsync，ブランキング信号nLBANKの記述です．

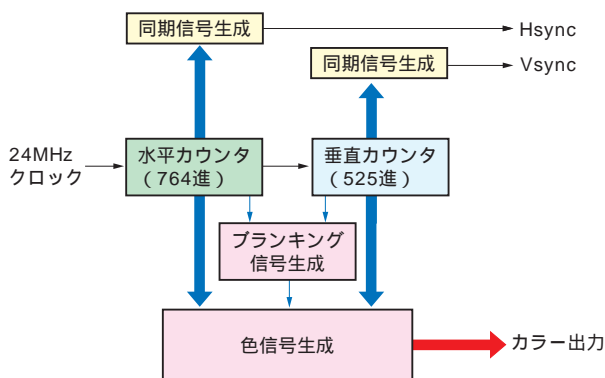



図6　VGA制御回路ブロック図
24MHzピクセル・クロックをベースに，水平同期カウンタ（764進），垂直同期カウンタ（525進）によりVGA表示のタイミング信号を生成する．

表4　カウンタ値によるタイミング設計
水平カウンタ，垂直カウンタの値をベースに各同期信号，ブランキング信号を生成する．

パラメータ	水平カウンタ		垂直カウンタ	
	開始値	終了値	開始値	終了値
周期	0	763	0	524
表示期間	0	639	0	479
ブランキング	640	763	480	524
同期パルス	640+16	640+16+63	480+10	480+10+1

**図5
RGBビデオ出力波形**
アナログRGB信号にブランキング（BLANK）信号，同期（SYNC）信号を含む場合のレベルを規定．

リスト1　同期信号，ブランキング信号の生成

```
signal clk24     : std_logic;
signal h_counter : std_logic_vector (9 downto 0);
signal v_counter : std_logic_vector (9 downto 0);

begin

process (clk48_in)
begin
    if clk48_in'event and clk48_in='1' then
          clk24<=not clk24;
    end if;
end process;

process (clk24)
begin
    if clk24'event and clk24='1' then
          if(h_counter<763) then
                h_counter<=h_counter+1;
          else
                h_counter<=(othes=>'0');
                if(v_counter<524) then
                      v_counter<=v_counter+1;
                else
                      v_counter<=(othes=>'0');
                end if;
          end if;
    end if;
end process;

Hsync='0' when h_counter>=640+16 and
          h_counter<640+16+96 else '1';
Vsync<='0' when v_counter>=480+10 and
          v_counter<480+10+2  else '1';
nLBLANK<='0' when h_counter>=640 or
          v_counter>=480 else '1';
```

① 内部信号の定義
② クロック分周により24MHzを生成
③ 764進カウンタ＋525進カウンタ
④ 水平同期信号，垂直同期信号，ブランキング信号の生成

図7　フランス国旗の色

水平同期カウンタの値に応じて色データを変えると，フランスの国旗ができる．

表5　フランス国旗の表示データ

水平カウンタの値に応じて表示色データを決定する．

水平カウンタ	表示色	赤(R)	緑(G)	青(B)
0～212	青	0	0	1
213～426	白	1	1	1
427～639	赤	1	0	0

リスト2　フランス国旗のカラー・ドット出力

```
signal rgb       : std_logic_vector(2 downto 0);

begin

rgb<="001" when h_counter<213  else
     "111" when h_counter<427  else
     "100";

LR<=(others=>rgb(2));
LG<=(others=>rgb(1));
LB<=(others=>rgb(0));
```

① 内部信号定義
② 水平カウンタの値に応じて，3原色を変化
③ 集合信号線に反映

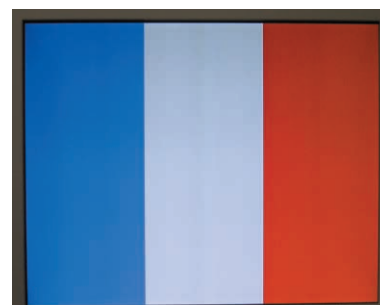
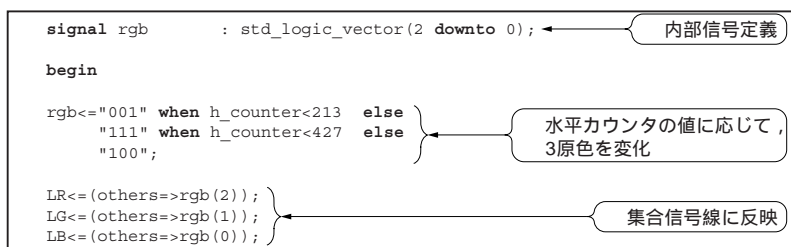



写真2　フランス国旗の表示

5. カラー信号の生成

まず簡単なカラー・パターンの例として，フランスの国旗を表示するカラー・ドット出力信号を生成してみましょう．フランス国旗はご存じのように，「自由」，「平等」，「博愛」を表す青，白，赤の領域が横に並んでいます（図7）．

これを640×480ドットの領域内で表示するには水平カウンタが0～639まで変化する途中で表示色を切り替えます．表5に示すように水平カウンタの値に応じて3原色をコントロールします．

VHDL記述をリスト2に示します．

①では3原色の状態を表す内部信号ベクトルrgbを定義しています．ビット並びの上位から，赤，緑，青の成分のあるなしを‘1’，‘0’で表現します．

②では水平カウンタの値によって表示色が青，白，赤になるようにrgbの各ビットを立てています．

③では，rgbの状態をR（赤），G（緑），B（青）各色の出力端子に反映しています．

たとえばrgb(2)が0ならLR5～LR0はすべて‘0’，rgb(2)が‘1’ならLR5～LR0はすべて‘1’となるようにしています．

全体のVHDLのソースをリスト3に示します．ビデオ用D-AコンバータADV7125の制御信号線については，

リスト3　フランス国旗の表示プログラム

```
----------------------------------------------------------
-- Engineer:       Masao Iwata
-- Create Date:    2007/05/05
-- Design Name:    france
-- Module Name:    CQ_VGA1 - Behavioral
----------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity CQ_VGA1 is
  port(clk48_in: in std_logic;
       LR       : out std_logic_vector(5 downto 0);
       LG       : out std_logic_vector(5 downto 0);
       LB       : out std_logic_vector(5 downto 0);
       nLSYNC   : out std_logic;
       nLBLANK  : out std_logic;
       LCLK     : out std_logic;
       nLPSAVE  : out std_logic;
       Hsync    : out std_logic;
       Vsync    : out std_logic
       );
end CQ_VGA1;

architecture Behavioral of CQ_VGA1 is

signal clk24      : std_logic;
signal h_counter : std_logic_vector(9 downto 0);
signal v_counter : std_logic_vector(9 downto 0);
signal hs,vs,bl   : std_logic;
signal rgb        : std_logic_vector(2 downto 0);


begin

process (clk48_in)
begin
  if clk48_in'event and clk48_in='1' then
               clk24<=not clk24;
  end if;
end process;

process (clk24)
begin
  if clk24'event and clk24='1' then
      if (h_counter<763) then
           h_counter<=h_counter+1;
      else
           h_counter<=(others=>'0');
           if (v_counter<524) then
                v_counter<=v_counter+1;
           else
                v_counter<=(others=>'0');
           end if;
      end if;
  end if;
end process;


hs<='0' when h_counter>=(640+16) and
             h_counter<(640+16+64) else '1';
vs<='0' when v_counter>=(480+10) and
             v_counter<(480+10+2)  else '1';
bl<='0' when h_counter>=640 or v_counter>=480 else '1';

nLSYNC<='1';
nLBLANK<=bl;
LCLK<=clk24;
nLPSAVE<='1';
Hsync<=hs;
Vsync<=vs;

rgb<="001" when h_counter<213  else
     "111" when h_counter<427  else     ←カラー信号生成
     "100";

LR<=(others=>rgb(2));
LG<=(others=>rgb(1));
LB<=(others=>rgb(0));

end Behavioral;
```

LCLK(クロック入力)←ピクセル・クロックCLK24

nLPSAVE，nLSYNC←常時High(インアクティブ)

となるようにしました．**写真2**は表示画面です．

6．8色カラー・バーの表示

次に**図8**の8色カラー・バーを表示させてみましょう．カラー・バーはカラー・テレビの調整用に用いられる標準表示画面です．アナログRGB回路，TFT液晶表示回路の場合も簡単にカラー信号の確認ができるので便利な画面です．

フランス国旗の表示(**リスト3**)のカラー信号生成記述部(灰色の部分)を**リスト4**に示すように変更するだけです．**写真3**は表示画面です．

7．市松模様の表示

次に**図9**の64×64ピクセルの白色・青色タイルによる市松模様を表示させてみましょう．これも**リスト3**の灰色の部分を変更するだけで簡単にできます．

水平，垂直カウンタの64の重みのある下から数えて第6ビット目の内容が‘0’，‘1’と変化するごとに色を変えればよいわけです．市松模様はパターンが水平，垂直両方向に交互に入れ替わっています．

水平カウンタの第6ビット目と垂直カウンタの第6ビット目とで排他的論理和をとって，その結果で色を決定するようにしました．**リスト3**の灰色の部分を**リスト5**のように書き換えます．**写真4**は表示画面です．

8．赤と緑による二次元グラデーションの表示

これまでの設計例は8色表示であり，画像ベースボードのD-Aコンバータが生かされていません．**図10**に示すように水平，垂直方向に原色の明るさを徐々に変化させてグラ

リスト4　8色カラー・バーのVHDL記述（リスト3の灰色の部分を変更）

```
rgb<= "000" when h_counter<80  else
      "001" when h_counter<160 else
      "010" when h_counter<240 else
      "011" when h_counter<320 else
      "100" when h_counter<400 else
      "101" when h_counter<480 else
      "110" when h_counter<560 else "111" ;
```

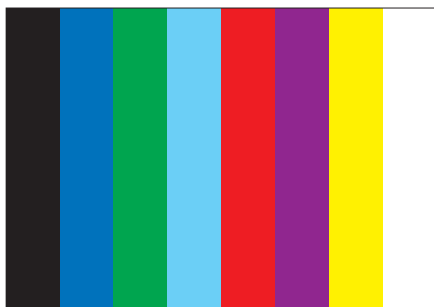



図8
8色カラー・バー
カラー・テレビの調整用に用いられる標準表示画面．色の発色を人目で確認できる．

写真3
8色カラー・バーの表示

リスト5　市松模様のVHDL記述（リスト3の灰色の部分を変更）

```
rgb<="111" when h_counter(6)='1' xor
v_counter(6)='1'  else "001";
```

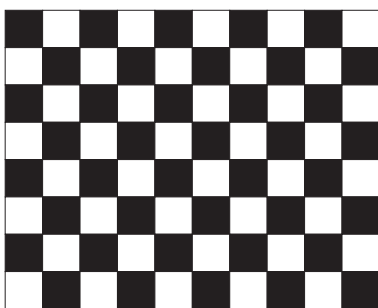



図9
市松模様
水平/垂直同期カウンタの6ビット目の変化に応じて色を変えるとできる．

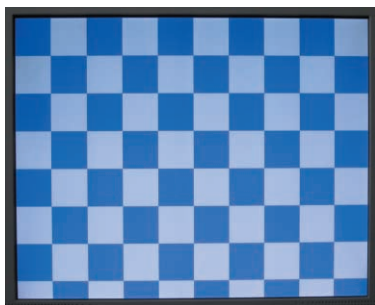



写真4
市松模様の表示

リスト6　2次元グラデーションのVHDL記述（リスト3の灰色の部分を変更）

```
LR<=h_counter(9 downto 5);
LG<=v_counter(9 downto 5);
LB<="10100";
```

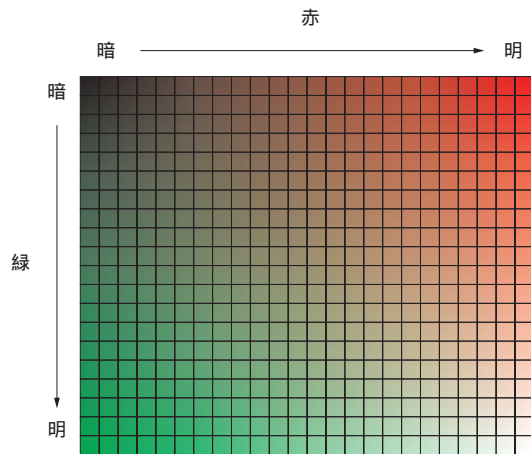



図10　2次元グラデーション
虹のような徐々に変化する模様．

写真5
2次元グラデーションの表示

デーションの表示をさせてみましょう．市松模様表示の応用です．

タイルの大きさは32×32ピクセルとします．水平方向の解像度は640ピクセル，垂直方向のそれは480ピクセルですから，横方向に20枚，縦方向には15枚のタイルが並びます．横方向に赤色の明るさを，縦方向に緑色の明るさを変化させます．青色の明るさは20で一定とします．

まず**リスト3**の内部信号rgbを削除し，**リスト6**に示すようにLR，LGに水平カウンタ，垂直カウンタの上位5ビットを代入します．

いかがでしょう．**写真5**に示すきれいなグラデーションが表示されたと思います．画像回路はVHDL記述の結果をすぐ目で確認できます．

緑と青，赤と青など組み合わせを変えるなど，お気に入

リスト7　秒カウンタのVHDL記述

```
-- Engineer:       Masao Iwata
-- Create Date:    2007/05/05
-- Design Name:    sec_counter
-- Module Name:    CQ_VGA2 - Behavioral
-- library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity CQ_VGA2 is
  port(clk48_in: in std_logic;
       LR      : out std_logic_vector(5 downto 0);
       LG      : out std_logic_vector(5 downto 0);
       LB      : out std_logic_vector(5 downto 0);
       nLSYNC  : out std_logic;
       nLBLANK : out std_logic;
       LCLK    : out std_logic;
       nLPSAVE : out std_logic;
       Hsync   : out std_logic;
       Vsync   : out std_logic
       );
end CQ_VGA2;

architecture Behavioral of CQ_VGA2 is

signal clk24      : std_logic;
signal h_counter : std_logic_vector(9 downto 0);
signal v_counter : std_logic_vector(9 downto 0);
signal hs,vs,bl   : std_logic;
signal rgb        : std_logic_vector(2 downto 0);
signal dot        : std_logic;
signal back_color: std_logic_vector(14 downto 0)
                 :="000000000001111";
signal segment    : std_logic_vector(7 downto 1)
                 :="1111011"; --9
signal cnt25      : std_logic_vector (24 downto 0);
signal cnt        : std_logic_vector (3 downto 0);

begin

process (clk48_in)
begin
  if clk48_in'event and clk48_in='1' then
             clk24<=not clk24;
end if;
end process;

process (clk24)
begin
  if clk24'event and clk24='1' then
      if (h_counter<763) then
           h_counter<=h_counter+1;
      else
           h_counter<=(others=>'0');
           if (v_counter<524) then
               v_counter<=v_counter+1;
           else
               v_counter<=(others=>'0');
           end if;
      end if;
  end if;
end process;


hs<='0' when h_counter>=(640+16) and
             h_counter<(640+16+64) else '1';
vs<='0' when v_counter>=(480+10) and
             v_counter<(480+10+2)  else '1';
bl<='0' when h_counter>=640 or v_counter>=480 else '1';

nLSYNC<='1';
nLBLANK<=bl;
LCLK<=clk24;
nLPSAVE<='1';
Hsync<=hs;
Vsync<=vs;

process (clk24)
begin
      if clk24'event and clk24='0' then
               if cnt25<23999999 then
                  cnt25<=cnt25+1;
               else
                 cnt25<=(others=>'0');
               end if;
      end if;
end process;

process (cnt25(24))
begin
      if cnt25(24)'event and cnt25(24)='0' then
               if cnt<9 then
                  cnt<=cnt+1;
               else
                  cnt<=(others=>'0');
               end if;
      end if;
end process;

process (cnt)
begin
       case cnt is
           when "0000" => segment <= "1111110";
           when "0001" => segment <= "0110000";
           when "0010" => segment <= "1101101";
           when "0011" => segment <= "1111001";
           when "0100" => segment <= "0110011";
           when "0101" => segment <= "1011011";
           when "0110" => segment <= "1011111";
           when "0111" => segment <= "1110000";
           when "1000" => segment <= "1111111";
           when "1001" => segment <= "1111011";
           when others => segment <= "1001000";
       end case;
end process;

dot <=
      '1' when segment(7)='1' and h_counter>=280+16
and h_counter<280+64 and v_counter>=196
and v_counter<196+12 else
      '1' when segment(6)='1' and h_counter>=280+68
and h_counter<280+80 and v_counter>=196+6
and v_counter<196+54  else
      '1' when segment(5)='1' and h_counter>=280+68
and h_counter<280+80 and v_counter>=196+58
and v_counter<196+106 else
      '1' when segment(4)='1' and h_counter>=280+16
and h_counter<280+64 and v_counter>=196+102
and v_counter<196+114 else
      '1' when segment(3)='1' and h_counter>=280
and h_counter<280+12 and v_counter>=196+58
and v_counter<196+106 else
      '1' when segment(2)='1' and h_counter>=280
and h_counter<280+12 and v_counter>=196+6
and v_counter<196+54  else
      '1' when segment(1)='1' and h_counter>=280+16
and h_counter<280+64 and v_counter>=196+50
and v_counter<196+62  else
      '0';

LR<="111111" when dot='1' else back_color(14 downto 10);
LG<="111111" when dot='1' else back_color(9 downto 5);
LB<="111111" when dot='1' else back_color(4 downto 0);

end Behavioral;
```

2

図12　秒カウンタの構成
24MHzのピクセル・クロックからカウンタにより秒信号をつくり，7セグメントで表示する．

24MHz → 1/24 000 000分周 → 10進カウンタ cnt → 7セグメント・デコーダ → カラー信号生成

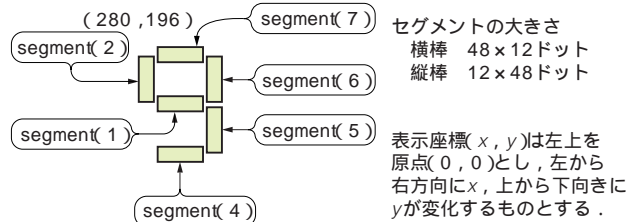
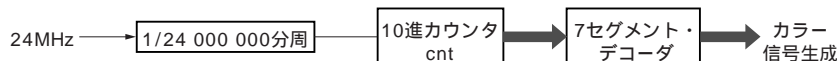
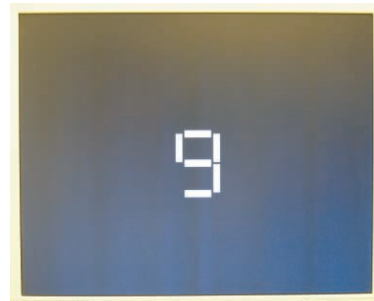



図11　数字表示レイアウト
画面上に7セグメントにより数字を表現する．

写真6　数字表示

りのグラデーション画面を作りながらVHDL記述の学習をしてみませんか．

9．数字の7セグメント表示と秒カウンタ

RGBディスプレイ画面上に**図11**に示すような7セグメントで数字を表示する回路を設計しました．

そして**図12**に示すように秒カウンタを設計し，表示してみました．**リスト7**はそのVHDL記述，**写真6**はその表示例です．

えさき・まさやす，いわた・まさお